

UPC for GPU Computing

Yili Zheng, Costin Iancu and Seung-Jai Min

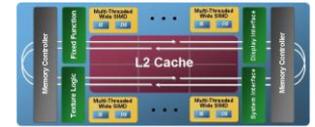
Future Technologies Group

Lawrence Berkeley National Laboratory

Pervasive GPU Computing

- Hardware:

- NVIDIA *Tesla*
- AMD *FireStream*
- Intel *Larabee*



- Software

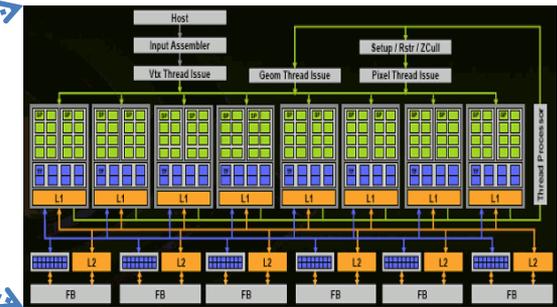
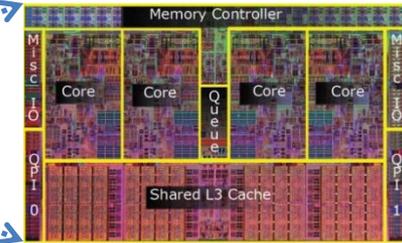
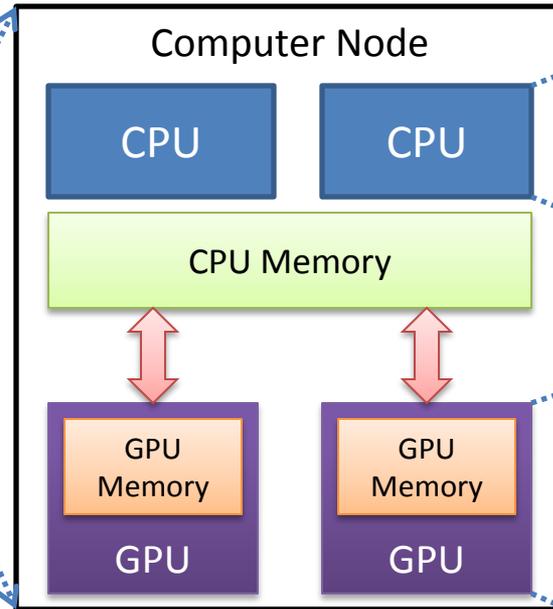
- CUDA
- Stream SDK
- DirectX Compute
- OpenCL



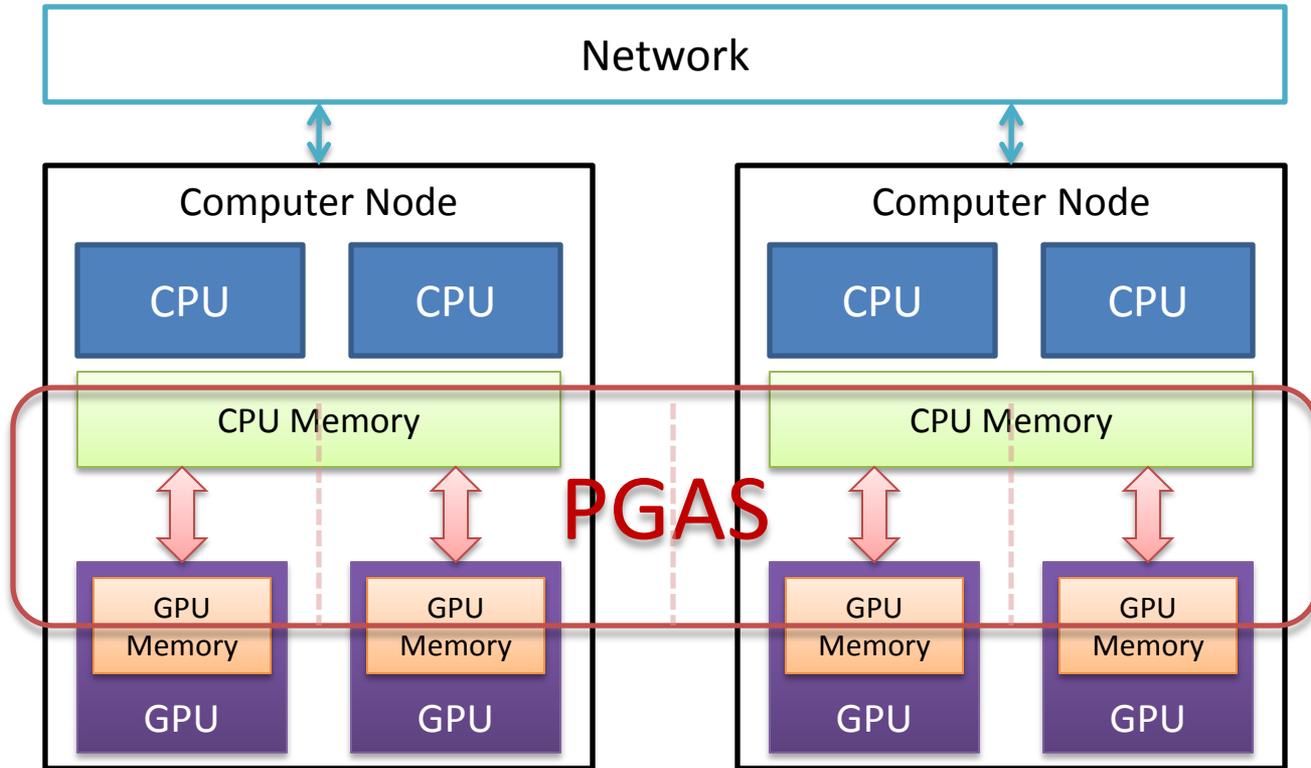
- Applications

- CUDA zone (~400 apps)
- Folding@Home
 - Top 2 architectures: NVIDIA GPU 1959 TFlops and ATI GPU 930 TFlops

GPU Computer Cluster



Partitioned Global Address Space



GPU and PGAS

- GPU programming
 - Small device memory, large host memory
 - Host to GPU communication explicitly programmed (aka DMA)
 - No GPU to GPU communication
- PGAS programming
 - Programming language support for data decomposition
 - Unified framework for data communication
- Advantages of using PGAS for GPU programming
 - Asynchronous parallel execution of CPU and GPU (see Cell work)
 - Automatic Host-to-GPU code generation
 - Using all the host memory for data intensive applications

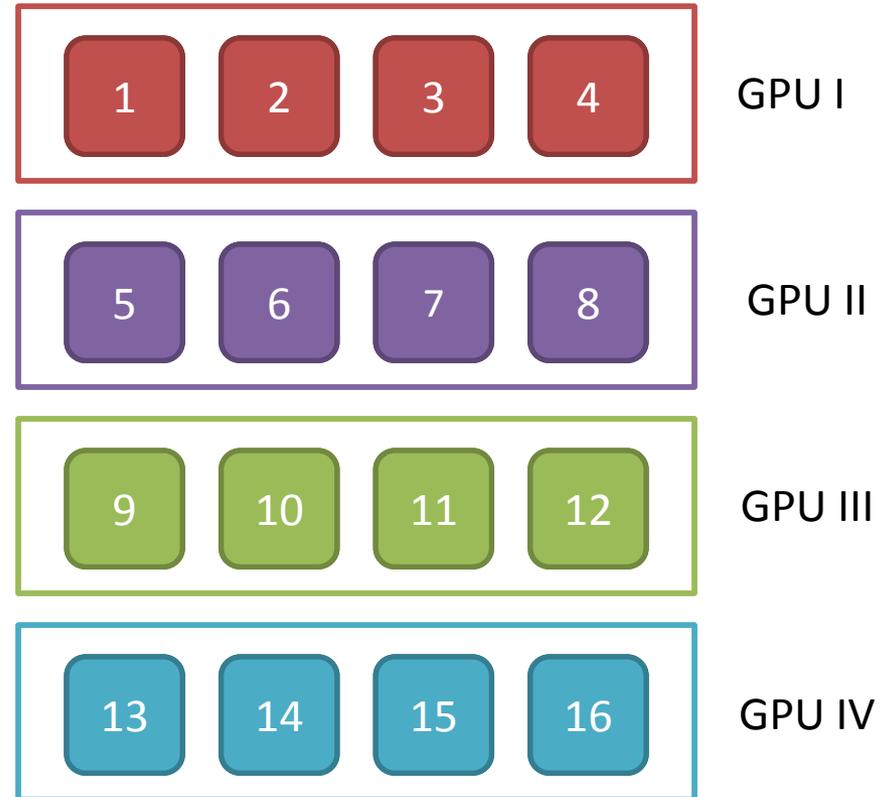
Fast Fourier Transform

Global Matrix View



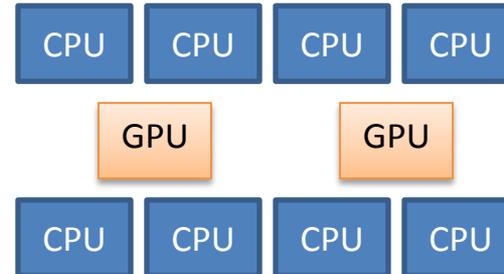
```
Shared [4] A[4][4];  
1D_FFT(A); // row fft  
A' = AT;  
1D_FFT(A'); // col fft  
A = A'T
```

Distributed Matrix Storage

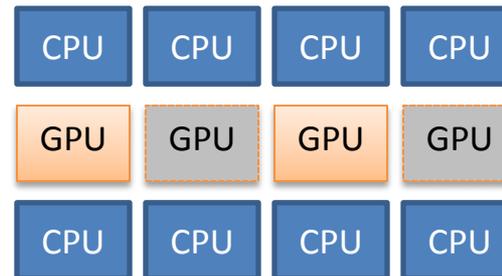


Execution Models

- Synchronous model



- Virtual GPU model

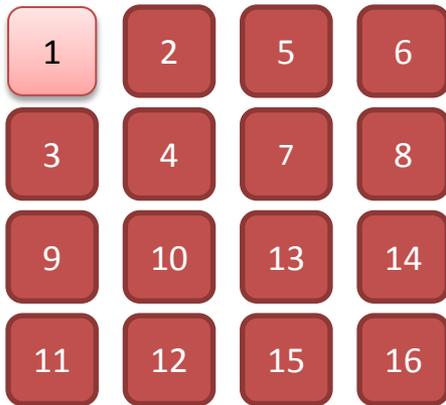


- Hybrid model

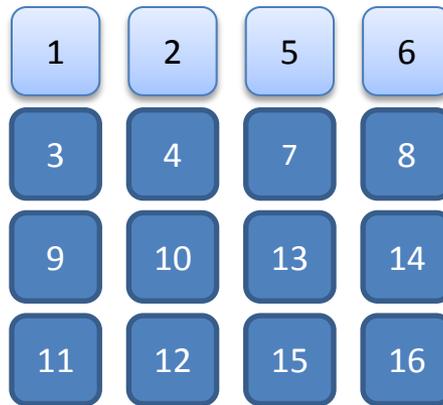


Parallel Matrix Multiplication

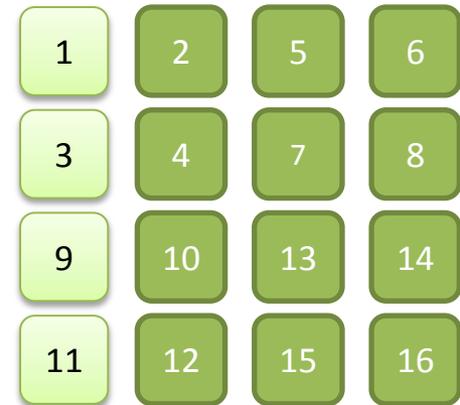
$$\begin{pmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{mn} \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \times \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mn} \end{pmatrix}$$



C



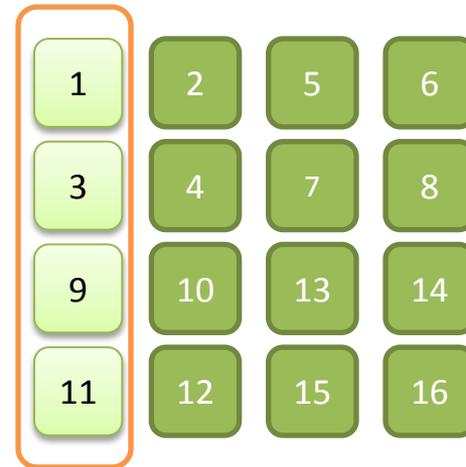
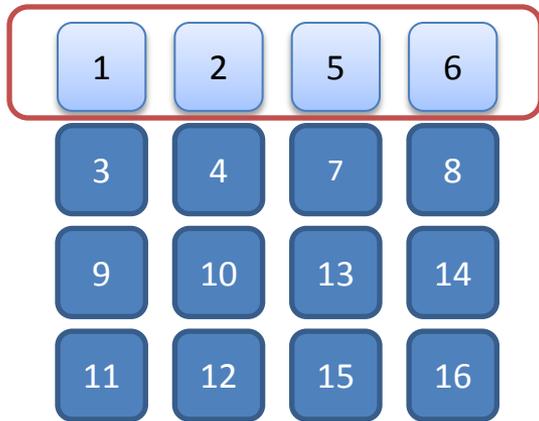
A



B

Collective Communication on GPUs

- Row team broadcast
- Column team broadcast

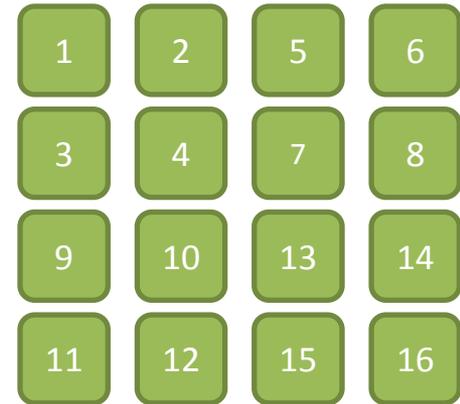
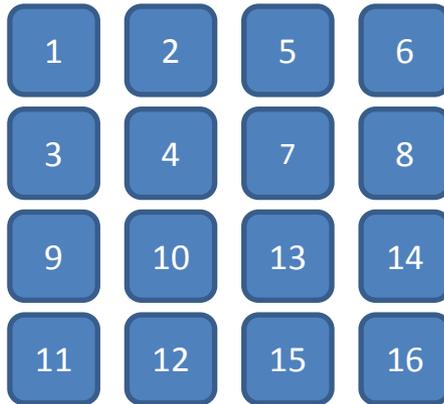
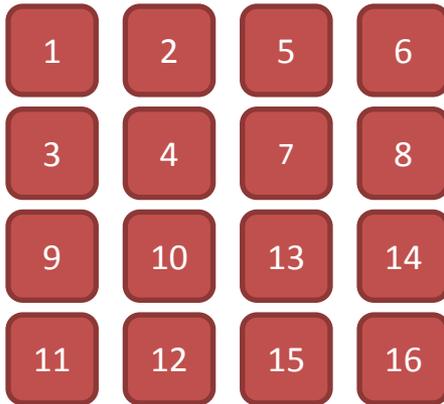


```
shared [2][2] float A[4][4], B[4][4], C[4][4];  
upc_team_bcast(A[1], row_team);  
upc_team_bcast(B[1], column_team);
```

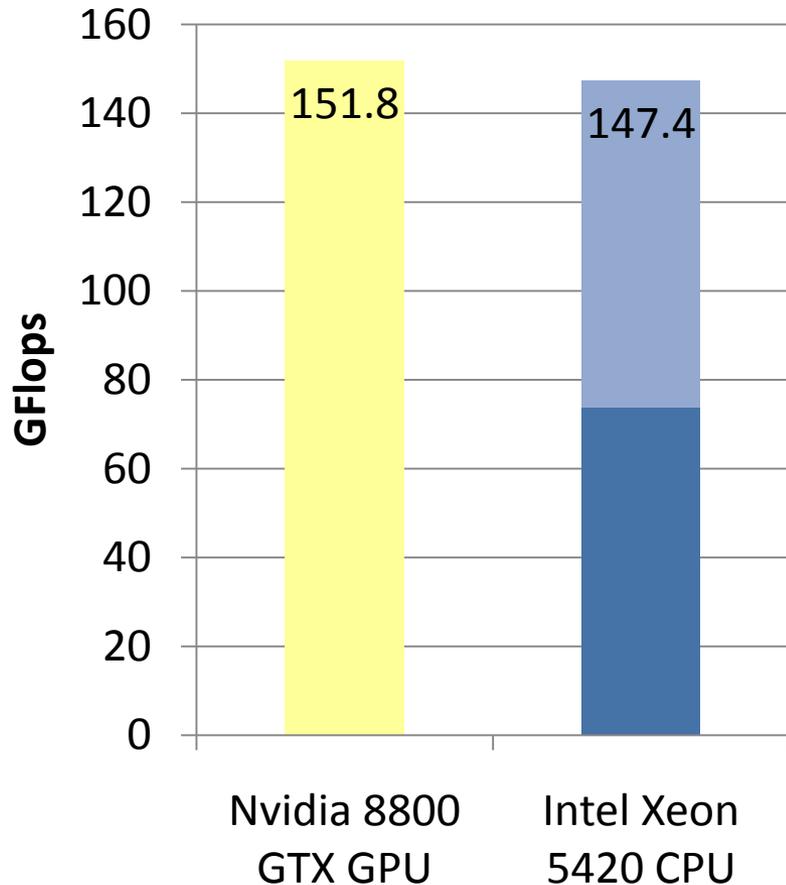
Write once run everywhere!

Data Tiling for GPU Computing

```
shared [2][2] float A[4][4], B[4][4], C[4][4];  
move_block_to_GPU(A[i][k], B[k][j], C[i][j]);  
cublasSgemm(A[i][k], B[k][j], C[i][j]);
```



GPU Matrix-Mult Experiment



- GPU computing
 - Matrices: 3 x 1GB
 - GPU memory: 768MB
 - 73% of CUDA BLAS peak
- CPU computing
 - 2 quad-core CPUs
 - Intel MKL
 - 91% of CPU peak
- Hybrid computing

Goals: PGAS for GPU Computing

- **Productivity**
 - Portable data decomposition and communication
 - Modular high performance libraries
 - Compiler translation from host language to GPU language and vice versa
- **Performance**
 - Locality optimizations
 - Runtime communication optimizations (automatic double buffering)
 - Use both host and GPU for computation
 - Use hybrid clusters (NxHost+MxGPU)

Research Areas

- Add global address space extension for “local stores” in the Berkeley UPC compiler
- Enable GPU-GPU communication in GASNet
- Develop “team” computation concepts
- Enhance language support for data layouts (e.g., 2-D block cyclic layout)
- Design algorithms for GPU clusters to use all available memory
- Collaborate with industry partners